CERTIFICATE OF TRANSMISSION BY FACSIMILE (37 CFR 1.8) Applicant(s): **Ricardo V. Martija et al**	Docket No. **APP 1208**

Application No.	Filing Date	Examiner	Group Art Unit
09/774,976	03/31/01	LAZARO, David R.	2155

Invention: **Method and System for Determining Geographical Regions of Hosts in a Network**

I hereby certify that this **Exhibit I (9 pages)**
(Identify type of correspondence)
is being facsimile transmitted to the United States Patent and Trademark Office (Fax. No. **(703) 872-9306**

on **April 29, 2005**
(Date)

Vivian Austin
(Typed or Printed Name of Person Signing Certificate)

Vivian Austin
(Signature)

Note: Each paper must have its own certificate of mailing.

```
#!/usr/local/bin/perl
# COPYRIGHT  (c) 1998 Bell Communications Research Inc.,
#   All Rights Reserved.
#
# PROPRIETARY - BELLCORE AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Bell Communications
# Research (Bellcore), and its authorized clients, except
# with written permission of Bellcore.
#
# $Id: getCovar.pl,v 1.2 1999/05/25 15:33:47 rmartija Exp rmartija $
#

undef;
require 'getopts.pl';
require '/u/rmartija/netsizer/scripts/math.pl';

$USAGE = "Usage: " . $0 . " [-D] -d domain  file\n\n" .
         "Options:\n" .
         "    -D          debug mode\n" .
         "    -d domain   domain type (1=US, 2=Non-US)\n" .
         "    file        name of input file. The default is STDIN.\n\n" .
         "Example:\n" .
         "   $0 ../data/test.out\n" .
         "   $0 -d 1 ../data/test.out\n" .
         "   $0 -D ../data/test.out\n" .
         "   $0 -D -d2 ../data/test.out\n\n" ;


##################################################################
####################### main program ############################
##################################################################

$x = &Getopts( 'd:D' );
die "$USAGE\n" unless ($x ne '');

die "$USAGE\n" unless $opt_d && $opt_d >= 1 && $opt_d <= 2;

if( $opt_d == 1 ) {
    $domain = 'US';
}
else {
    $domain = 'NONUS';
}

$oldLoc = '';
$rows = 0;
$cols = 0;

die "$USAGE\n" if( $#ARGV > 0 );

if( $#ARGV < 0 || $ARGV[0] eq '-' ) {
    $INPUT = STDIN;
}
else {
        die "ERROR: cannot open $ARGV[0]\n" unless -r $ARGV[0];
        open( INPUT, "< $ARGV[0]" );
        $INPUT = INPUT;
    }

    while( <$INPUT> ) {
        chop;
        next unless length($_) > 0;
        @tokens = split( '\t', $_ );
        $locale = $tokens[0];

        if( $locale ne $oldLoc ) {
            if( $oldLoc ne '' ) {
                %m = &getMeans( $rows-1, $cols, *matrix );
                print "$domain: $oldLoc\n";
                print "MEAN: " ;
                for( $i = 1; $i <= $cols; $i++ ) {
                    printf "%.2f", $m{$i} ;
                    print " " if( $i < $cols );
                    print "\n" if( $i == $cols );
                }

                if( $opt_D ) {
                    print "ORIGINAL MATRIX:\n" ;
                    for( $i = 1; $i <= $cols; $i++ ) {
                        for( $j = 1; $j <= $cols; $j++ ) {
                            printf "%12.2f", $matrix{$j + (($i - 1) * $cols)} ;
                            print " " if( $j < $cols );
                            print "\n" if( $j == $cols );
                        }
                    }
                    print "\n" ;
                }

                %S = &getCovarianceMatrix(  $rows-1, $cols, *matrix, *m );
                if( $opt_D ) {
                    print "COVARIANCE MATRIX:\n" ;
                    for( $i = 1; $i <= $cols; $i++ ) {
                        for( $j = 1; $j <= $cols; $j++ ) {
                            printf "%12.2f", $S{$j + (($i - 1) * $cols)} ;
                            print " " if( $j < $cols );
                            print "\n" if( $j == $cols );
                        }
                    }
                    print "\n" ;
                }

                %I = &getInverseMatrix( $cols, *S );
                print "INVERSE OF COVARIANCE MATRIX:\n" ;
                for( $i = 1; $i <= $cols; $i++ ) {
                    for( $j = 1; $j <= $cols; $j++ ) {
                        printf "%12.2f", $I{$j + (($i - 1) * $cols)} ;
                        print " " if( $j < $cols );
                        print "\n" if( $j == $cols );
                    }
                }
                print "\n";
            }

            $oldLoc = $locale;
            $rows = 1;
            $cols = @tokens - 1;
        }

        for( $j = 1; $j <= $cols; $j++ ) {
            $matrix{$j + (($rows - 1) * $cols)} = $tokens[$j] * 1.0;
        }

        $rows++;
    }

    close( $INPUT ) unless $#ARGV < 0 || $ARGV[0] eq '-';

    %m = &getMeans( $rows-1, $cols, *matrix );
    print "$domain: $oldLoc\n";
    print "MEAN: " ;
    for( $i = 1; $i <= $cols; $i++ ) {
        printf "%.2f", $m{$i} ;
        print " " if( $i < $cols );
        print "\n" if( $i == $cols );
    }

    if( $opt_D ) {
        print "ORIGINAL MATRIX:\n" ;
        for( $i = 1; $i <= $cols; $i++ ) {
            for( $j = 1; $j <= $cols; $j++ ) {
                printf "%12.2f", $matrix{$j + (($i - 1) * $cols)} ;
                print " " if( $j < $cols );
                print "\n" if( $j == $cols );
            }
        }
        print "\n" ;
    }

    %S = &getCovarianceMatrix(  $rows-1, $cols, *matrix, *m );
    if( $opt_D ) {
        print "COVARIANCE:\n" ;
        for( $i = 1; $i <= $cols; $i++ ) {
            for( $j = 1; $j <= $cols; $j++ ) {
                printf "%12.2f", $S{$j + (($i - 1) * $cols)} ;
                print " " if( $j < $cols );
                print "\n" if( $j == $cols );
            }
        }
        print "\n" ;
    }

    %I = &getInverseMatrix( $cols, *S );
    print "INVERSE OF COVARIANCE MATRIX:\n" ;
    for( $i = 1; $i <= $cols; $i++ ) {
        for( $j = 1; $j <= $cols; $j++ ) {
            printf "%12.2f", $I{$j + (($i - 1) * $cols)} ;
            print " " if( $j < $cols );
            print "\n" if( $j == $cols );
```

Exhibit II

```
#!/usr/local/bin/perl
# COPYRIGHT  (c) 1998 Telcordia Technologies Inc.,
#   All Rights Reserved.
#
# PROPRIETARY - BELLCORE AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia Technologies
# (Telcordia), and its authorized clients, except
# with written permission of Telcordia.
#
# $Id: getHostLoc.pl,v 1.1 1999/05/20 22:27:07 rmartija Exp rmartija $
#

require 'getopts.pl' ;

undef;

$USAGE = "Usage: " . $0 . " [-D] -u file -m file\n" .
         "Flags:\n" .
         "   -D          debug mode\n" .
         "   -u file     file containing the list of unclassified IP\n" .
         "               addresses (i.e. those with unknown locations)\n" .
         "               and their characteristics.\n" .
         "   -m file     file containing the means and inverse of covariance\n" .
         "               matrices\n" .
         "Examples:\n" .
         "   $0 -u unknowns -m matrix" ;


%g_means = ();
%g_inverse = ();
@g_locales;
$g_debug;
$g_attributes;


#----------------------------------------------------------------
#----------------------------------------------------------------
sub getDistance {
    my( $loc, $data ) = @_;

    my( @X ) = @$data;

    my( @mu ) = @{$g_means{$loc}};
    my( @sigma ) = @{$g_inverse{$loc}};

    my( @diff, @prod );
    my( $i, $j );

    for( $i = 0; $i <= $g_attributes; $i++ ) {
        $diff[$i] = $mu[$i] - $X[$i];
    }

    #
    # compute diff(transpose) * sigma. diff(transpose) is a 1 x N matrix
    # and sigma is a N x N matrix. the result is a 1 x N matrix.
    #
    for( $i = 0; $i <= $g_attributes; $i++ ) {
        $prod[$i] = 0.;
        for( $j = 0; $j <= $g_attributes; $j++ ) {
            $prod[$i] += $diff[$j] * $sigma[$i][$j];
        }
    }

    #
    # multiply the matrix obtained above, i.e prod, with diff. prod is a
    # a 1 x N matrix and diff is a N x 1 matrix. the result is a scalar.
    #
    my( $dist ) = 0;
    for( $i = 0; $i <= $g_attributes; $i++ ) {
        $dist += $prod[$i] * $diff[$i];
    }

    return $dist;
}


#----------------------------------------------------------------------
#----------------------------------------------------------------------
sub readMeansAndMatrices {
    my( $file ) = @_;

    open( F, "< $file" );
    @lines = <F>;
    close( F );

    my( $n_rows, $cur_row, $line_num ) = (-1, 0, 0);
    my( $cur_loc, $n_means );

    foreach( @lines ) {
        chop;

        $line_num++;

        next if $_ =~ /^\s*$/;    # skip blank lines

        if( $_ =~ /^US.*:\s*(.*)/ ) {
            die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                unless $n_rows < 0;

            # $1 contains the state string (e.g. NJ)
            $cur_loc = "$1,US";
            $cur_row = 0;
        }
        elsif( $_ =~ /^NONUS.*:\s*(.*)/ ) {
            die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                unless $n_rows < 0;

            # $1 contains the country string (e.g. BE)
            $cur_loc = "$1,$1";
            $cur_row = 0;
        }
        elsif( $_ =~ /^MEAN.*:\s*(.*)/ ) {
            die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                unless $n_rows < 0;

            # $1 contains something like 18.43 1130.71 20.00 170.71 19.57 228.5
            my( @means ) = split( ' ', $1 );
            $n_means = $n_rows = $#means;
            $g_means{$cur_loc} = \@means;
        }
        elsif( $_ =~ /^INVERSE.*:\s*(.*)/ ) {
            die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                unless $cur_row == 0;
        }
        elsif( $_ =~ /^([A-Za-z]+).*:/ ) {
            die "ERROR: Invalid Tag in $file\n-> line $line_num: $_\n";
        }
        else {
            my( @row ) = split( ' ', $_ );

            # make sure the matrix is a $n_means X $n_means array
            die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                unless $#row == $n_means && $cur_row <= $n_means;

            my( $r_entry ) = [@row];
            push( @{$g_inverse{$cur_loc}}, $r_entry );

            $cur_row++;
            $n_rows--;
        }
    }

    die "ERROR: $file is corrupted. More data expected.\n" unless $n_rows < 0;

    @g_locales = keys %g_means;
    return $n_means;
}

#----------------------------------------------------------------
#----------------------------------------------------------------
sub classifyIPs {
    my( $file ) = @_;

    open( F, "< $file" );

    my( @data, $tloc, $loc, %dist, $min );

    while( <F> ) {
        chop;
        next unless $_ =~ /^(\d+)\.(\d+)\.(\d+)\.(\d+).*\s*(.*)/;

        ($ip, @data) = split( ' ' );

        next unless $#data == $g_attributes;

        $min = time;    # initialize $dist to some arbitrary large number
                        # such as the number of seconds since 1/1/1970
        foreach $tloc ( @g_locales ) {
            $dist{$tloc} = &getDistance( $tloc, \@data );
            if( $dist{$tloc} < $min ) {
                $min = $dist{$tloc};
                $loc = $tloc;
            }
        }

        if( $g_debug ) {
            foreach $key (sort keys %dist) {
                printf "%-15s   %-8s   %7.2f\n", $ip, $key, $dist{$key};
            }
        }

        printf "%-15s %-8s\n", $ip, $loc;
    }

    close( F );
}


#################################################################
######################## main program ############################
#################################################################

$x = &Getopts( 'u:m:D' );
die "$USAGE\n" unless ($x ne '');
die "$USAGE\n" unless ($opt_u && $opt_m) ;

die "ERROR: cannot open $opt_u\n" unless -e $opt_u;
die "ERROR: cannot open $opt_m\n" unless -e $opt_m;

$g_debug = 1 if( $opt_D );
$g_attributes = &readMeansAndMatrices( $opt_m );
&classifyIPs( $opt_u );
```

Exhibit I11

```
#!/usr/local/bin/perl
# COPYRIGHT  (c) 1998 Bell Communications Research Inc.,
#   All Rights Reserved.
#
# PROPRIETARY - BELLCORE AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Bell Communications
# Research (Bellcore), and its authorized clients, except
# with written permission of Bellcore.
#
# $Id: classify.pl,v 1.1 1999/05/05 13:20:39 rmartija Exp $
#

undef;
require 'getopts.pl';

$USAGE = "Usage: " . $0 . " [-D] -p path -d file -t type -h file\n\n" .
        "Options:\n" .
        "    -D          debug mode\n" .
        "    -p path     output directory\n" .
        "    -d file     name of domains file\n" .
        "    -t type     domain type (1=US only, 2=Non-US only, 3=Global)\n" .
        "    -h file     name of hosts file\n\n" .
        "Example:\n" .
        "   $0 -p ../data/local -d alldomains.lcl -t 1 -h dat.txt\n" .
        "   $0 -D -p ../data/national -d alldomains.nat -t 2 -h dat.txt\n" .
        "   $0 -p ../data/non-us -d alldomains.nus -t 3 -h dat.txt\n\n" ;


#-----------------------------------------------------------------
#-----------------------------------------------------------------
sub prompt {
    my( $msg, $choices, $default, $nocase ) = @_;
    my( $reply );

    print STDERR $msg;

    print STDOUT " ($choices)" if( $choices );
    print STDOUT "? ";
    print STDOUT "[$default] " if( $default );

    $reply = <STDIN>;
    chop( $reply );

    ($reply =~ tr/a-z/A-Z/) if( $nocase );
    ($reply = $default) if( length($reply) == 0 && $default );

    return $reply;
}


#-----------------------------------------------------------------
#-----------------------------------------------------------------
sub overwrite {
    my( $file ) = @_;
```

```
        return 1 if( ! (-e $file || -d $file) );

        my( $msg ) = "$file exists. Overwrite";
        return 1 if( &prompt($msg, "Y/N", "Y", 1) eq "Y" );
        return 0;
    }

    ####################################################################
    ####################### main program ##############################
    ####################################################################

    $x = &Getopts( 'd:h:p:t:D' );
    die "$USAGE\n" unless ($x ne '');

    die "ERROR: cannot open $opt_d\n" unless -r $opt_d;
    die "ERROR: cannot open $opt_h\n" unless -r $opt_h;
    die "ERROR: domain type not specified\n$USAGE\n" unless $opt_t;

    die "ERROR: invalid domain type ($opt_t)\n$USAGE\n"
        unless (int($opt_t) >= 1 && int($opt_t) <= 3);

    chop( $basket = `basename $opt_d` ) ;
    open( F, "< $opt_d" ) ;
    @domains = <F> ;
    close( F ) ;

    unlink "$opt_p/$basket" unless $opt_t == 3;
    foreach $domainName (@domains) {
        chop( $domainName ) ;
        $cmd = "grep \'\\.$domainName\$\' $opt_h | cut -d\":\" -f1 " ;

        if( $opt_t == 3 ) {
            $cmd .= "> $opt_p/$domainName" ;
        }
        else {
            $cmd .= ">> $opt_p/$basket" ;
        }

        `$cmd`;
    }
```